

Parameterized Verification of Graph Transformation Systems with Whole Neighbourhood Operations^{*}

Giorgio Delzanno¹ and Jan Stückrath²

¹ Univerità di Genova, Italy
giorgio.delzanno@unige.it

² Universität Duisburg-Essen, Germany
jan.stueckrath@uni-due.de

Abstract. We introduce a new class of graph transformation systems in which rewrite rules can be guarded by universally quantified conditions on the neighbourhood of nodes. These conditions are defined via special graph patterns which may be transformed by the rule as well. For the new class for graph rewrite rules, we provide a symbolic procedure working on minimal representations of upward closed sets of configurations. We prove correctness and effectiveness of the procedure by a categorical presentation of rewrite rules as well as the involved order, and using results for well-structured transition systems. We apply the resulting procedure to the analysis of the Distributed Dining Philosophers protocol on an arbitrary network structure.

1 Introduction

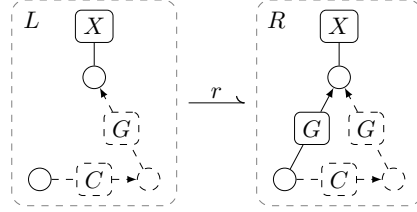
Parameterized verification of distributed algorithms is a very challenging task. Distributed algorithms are often sensible to the network topology and they are based on communication patterns like broadcast messages and conditions on channels that can easily generate undecidable verification instances or finite-state problems of high combinatorial complexity. In order to naturally model interaction rules of topology-sensitive protocols it seems natural to consider languages based on graph rewriting and transformations as proposed in [21]. However, in this formalism rules can only match fixed subgraph in the graph they are applied to. Since we need to specify rules where the entire neighbourhood of a node is matched by the rule, we extend the standard approach by universally quantified patterns attached to nodes. With these patterns the matching of a left side of a rule can be increased until the entire neighbourhood of a node is covered. If the matching cannot be extended in this way the rule is not applicable, e.g. we could formalize a rule which only matches a node when every incident edge is incoming. Additionally the matched occurrences of the patterns can also be changed by the rule. A similar approach are adaptive star grammars [20], the difference being that we do not restrict our left rule sides to be stars.

The resulting formal language can be applied to specify distributed versions of concurrent algorithms like Dining Philosophers in which neighbour processes use channels

^{*} Research partially supported by DFG project GaReV. This paper is an extended version of [17] additionally containing the proofs.

to request and grant access to a given shared resource. The protocol we use has been proposed by Namjoshi and Treller in [26]. There requests are specified using process identifiers attached to edges representing point-to-point communication channels. Universally quantified guards are used to ensure mutual exclusive access to a resource. In this paper we formulate the protocol without need of introducing identifiers. We instead use our extended notion of graph transformation systems to specify ownership of a given communication link. Universally quantified patterns attached to a requesting node are used then as guards to ensure exclusive access. Erroneous or undesirable configurations in the algorithm can be presented by a set of minimal error configurations. We then use a backward procedure to check if a configuration containing one of the error configurations is reachable. If none is reachable, the algorithm is proven to be correct.

Following the approach proposed in [7,25], we use basic ingredients of graph transformation and category theory (e.g. pushouts) to formally specify the operational semantics of our model. Parameterized verification for the resulting model is undecidable in general, even without universally quantified patterns [7]. To overcome this problem, we provide an approximated symbolic backward procedure using result for well-structured transition systems [6,22] to guarantee correctness and termination. Although the over-approximation is based on the monotonic abstraction approach proposed in [3,5], its application to the considered class of infinite-state systems is highly non trivial. In fact, our universal quantification approach is not restricted to process states only, but it can specify complex graph patterns as shown on the right. There the node marked with the X -edge represents a group where every node attached with a G -edge is a member of.



The rule can be applied if every edge attached to the two solid nodes is matched and has the form of the dashed part (the quantification). Effectively the rule adds a node to a group if all other connected nodes (via a C -edge) are already members of the group.

We have implemented a prototype version of the algorithms in the tool UNCOVER and tested on some case-studies. For instance, our prototype can verify the Distributed Dining Philosophers example without need of additional invariants as in [26]. Due to space limitations, the proofs can be found in Appendix B.

2 Preliminaries

In this paper we use hypergraphs, a generalization of directed graphs, where an edge can connect an arbitrary large but finite set of nodes. Furthermore we use graph morphisms to define rewriting rules.

Hypergraph Let Λ be a finite sets of edge labels and $ar: \Lambda \rightarrow \mathbb{N}$ a function that assigns an arity to each label (including the arity zero). A (Λ) -hypergraph (or simply graph) is

a tuple (V_G, E_G, c_G, l_G) where V_G is a finite set of nodes, E_G is a finite set of edges, $c_G : E_G \rightarrow V_G^*$ is a connection function and $l_G : E_G \rightarrow \Lambda$ is an edge labelling function. We require that $|c_G(e)| = ar(l_G(e))$ for each edge $e \in E_G$. An edge e is called *incident* to a node v if v occurs in $c_G(e)$. An *undirected path* of length n in a hypergraph is an alternating sequence $v_0, e_1, v_1, \dots, v_{n-1}, e_n, v_n$ of nodes and edges such that for every index $1 \leq i \leq n$ both nodes v_{i-1} and v_i are incident to e_i and the undirected path contains all nodes and edges at most once.

Let G, G' be (Λ) -hypergraphs. A *partial hypergraph morphism* (or simply *morphism*) $\varphi : G \rightarrow G'$ consists of a pair of partial functions $(\varphi_V : V_G \rightarrow V_{G'}, \varphi_E : E_G \rightarrow E_{G'})$ such that for every $e \in E_G$ it holds that $l_G(e) = l_{G'}(\varphi_E(e))$ and $\varphi_V(c_G(e)) = c_{G'}(\varphi_E(e))$ whenever $\varphi_E(e)$ is defined. Furthermore if a morphism is defined on an edge, it must be defined on all nodes incident to it. We denote *total morphisms* by an arrow of the form \rightarrow and write \succrightarrow if the total morphism is known to be injective.

Pushout Our rewriting formalism is the so-called *single-pushout approach* (SPO) based on the categorical notion of pushouts in the category of graphs and partial graph morphisms [21]. Given two morphisms $\varphi : G_0 \rightarrow G_1$ and $\psi : G_0 \rightarrow G_2$, the *pushout* of φ, ψ consists of the graph G_3 and two morphisms $\varphi' : G_1 \rightarrow G_3$ and $\psi' : G_2 \rightarrow G_3$. It corresponds to a merge of G_1 and G_2 along a common interface G_0 while at the same time deleting every element of one of the graphs if it has a preimage in G_0 which is not mapped to an element in the other graph. It is known that in our category the pushout of two morphisms always exists and is unique (up to isomorphism). It can be computed in the following way.

Let \equiv_V and \equiv_E be the smallest equivalences on $V_{G_1} \cup V_{G_2}$ and $E_{G_1} \cup E_{G_2}$ satisfying $\varphi(v) \equiv_V \psi(v)$ for all $v \in V_{G_0}$ and $\varphi(e) \equiv_E \psi(e)$ for all $e \in E_{G_0}$. The nodes and edges of the pushout object G_3 are then all *valid* equivalence classes of \equiv_V and \equiv_E . An equivalence class is *valid* if it does not contain the image of some $x \in G_0$ for which $\varphi(x)$ or $\psi(x)$ is undefined. The equivalence class of an edge is also considered invalid if it is incident to a node with an invalid equivalence class. The morphisms φ' and ψ' map each element to its equivalence class if this class is valid and are undefined otherwise.

For a backward step in our procedure we also need the notion of a *pushout complement* which is, given $\varphi : G_0 \rightarrow G_1$ and $\psi' : G_1 \rightarrow G_3$, a graph G_2 and morphisms $\psi : G_0 \rightarrow G_2, \varphi' : G_2 \rightarrow G_3$ such that G_3 is the pushout of φ, ψ . For graphs pushout complements not necessarily exist and if they exist there may be infinitely many. See [23] for a detailed description on how pushout complements can be computed.

GTS A *rewriting rule* is a partial morphism $r : L \rightarrow R$, where L is called left-hand and R right-hand side. A *match* (of r) is a total and injective morphism $m : L \rightarrow G$. Given a rule and a match, a *rewriting step* or rule application is given by a pushout diagram as shown on the right, resulting in the graph H . Note that injective matchings are not a restriction since non-injective matchings can be simulated, but are necessary for universally the quantified rules defined later.

$$\begin{array}{ccc} L & \xrightarrow{r} & R \\ m \downarrow & & \downarrow m' \\ G & \longrightarrow & H \end{array}$$

A *graph transformation system (GTS)* is a finite set of rules \mathcal{R} . Given a fixed set of graphs \mathcal{G} , a *graph transition system* on \mathcal{G} generated by a graph transformation system \mathcal{R} is represented by a tuple $(\mathcal{G}, \Rightarrow)$ where \mathcal{G} is the set of states and $G \Rightarrow G'$ if and only if $G, G' \in \mathcal{G}$ and G can be rewritten to G' using a rule of \mathcal{R} .

A computation is a sequence of graphs G_0, G_1, \dots s.t. $G_i \Rightarrow G_{i+1}$ for $i \geq 0$. G_0 can reach G_1 if there exists a computation from G_0 to G_1 .

3 Graph Transformations with Universally Quantified Conditions

To clarify the ideas and illustrate the usefulness of universally quantified conditions on the neighbourhood of nodes, let us consider the following example.

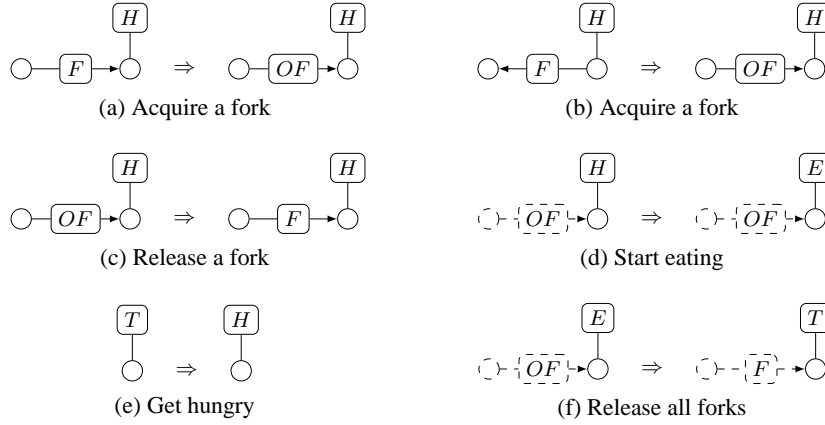


Fig. 1: Modelling of the dining philosophers problem on an arbitrary net

Example 1. Figure 1 shows a set of rules describing the Dining Philosophers Problem on an arbitrary graph structure. Each node represents a philosopher who can be in one of three different states: hungry (H), eating (E) or thinking (T). Each state is indicated by a unary edge attached to the philosopher. Between two philosophers there may be a free fork (an F-edge) or a fork owned by one of the philosophers (an OF-edge pointing to its owner). Note that our directed edges are in fact hyperedges of arity two, where the first node is the source and the second node is the target.

Philosophers can take unowned forks (Figure 1a and 1b) and also release control (Figure 1c). If a philosopher owns all connected forks, he can start to eat (Figure 1d). The dashed part of the rule indicates a universal quantification, meaning that the rule can only be applied if all edges attached to the philosopher are part of the matching and in fact forks owned by him. At some point the philosopher finished eating, releasing all forks (Figure 1f) and may become hungry in the future (Figure 1e). When releasing all forks, all forks owned by the philosopher are converted to unowned forks.

Rules matching the entire neighbourhood of a node (in the following called *quantified node*), such as the rules in Figure 1d and 1f cannot be described by normal rewriting rules. Therefore we extend normal rules to so-called universally quantified rules consisting of a normal rule and a set of universal quantifications. The idea is to first find a matching for the rule and then extend the rule as well as the matching until the entire neighbourhood of quantified nodes is part of the matching.

We apply the rule in Figure 1f to the graph G shown in Figure 2. There exists a match $m : L \rightarrowtail G$ where $r : L \rightarrow R$ is the rule without any use of the quantification. However, this matching does not match the entire neighbourhood of the quantified node (marked grey). Before applying the rule we have to add multiple copies of the quantification to r generating a so-called instantiation η where the extended match \overline{m} contains the entire neighbourhood of the quantified node.

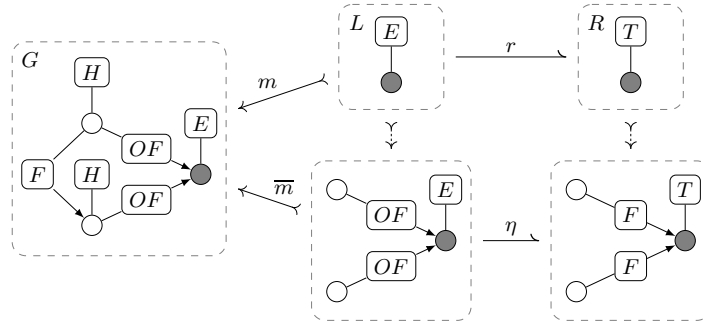


Fig. 2: A match of a universally quantified rule has to be extended until the entire neighbourhood of each quantified node is matched

In the following we formalize the notion of universally quantified rules as an extension of normal rules and introduce instantiations via a sequence of recursive instantiation steps.

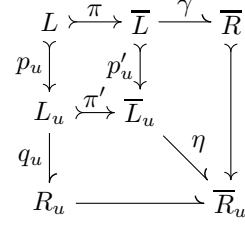
Definition 1 (Universally quantified rules). A universally quantified rule is a pair $\rho = (r, U)$, where $r : L \rightarrow R$ is a partial morphism and U is a finite set of universal quantifications. A universal quantification is a pair $(p_u, q_u) = u \in U$ where $p_u : L \rightarrowtail L_u$ is a total injective morphism and $q_u : L_u \rightarrow R_u$ is a partial morphism satisfying the restriction that $q_u(p_u(x))$ is defined and has exactly one preimage in L_u for every $x \in L$.

With $qn(u)$ we denote the set of quantified nodes of u , which is the set of all $v \in V_L$ such that there is an edge incident to $p_u(v)$ which has no preimage in L . We denote the quantified nodes of a rule the same way, i.e. $qn(\rho) = \bigcup_{u \in U} qn(u)$. We require that $qn(u) \neq \emptyset$ for all $u \in U$.

In the rest of the paper we will use UGTS to denote the extension of GTS with universally quantified rules.

Definition 2 (Instantiation of a universally quantified rule). An instantiation of a universally quantified rule $\rho = (r, U)$ consists of a total injective morphism $\pi : L \rightarrow \bar{L}$ and a partial morphism $\gamma : \bar{L} \rightarrow \bar{R}$ and is recursively defined as follows:

- The pair $(id_L : L \rightarrow L, r)$, where id_L is the identity on L , is an instantiation of ρ .
- Let $(\pi : L \rightarrow \bar{L}, \gamma : \bar{L} \rightarrow \bar{R})$ be an instantiation of ρ and let $(p_u : L \rightarrow L_u, q_u : L_u \rightarrow R_u) = u \in U$. Furthermore, let \bar{L}_u be the pushout of π, p_u and let \bar{R}_u be the pushout of $\gamma \circ \pi, q_u \circ p_u$, as shown in the diagram to the right. Then $p'_u \circ \pi$ and the (unique) mediating morphism η are also an instantiation of ρ . We write $(p'_u \circ \pi, \eta) = (\pi, \gamma) \diamond u$ to indicate that the instantiation (π, γ) was extended by u .



We say that the length of an instantiation is the number of steps performed to generate the instantiation, where (id_L, r) has a length of 0.

Example 2. Figure 3 shows a possible instantiation of the rule in Figure 1f. There is only one universal quantification u and this quantification is used once to generate the instantiation $(p'_u \circ id_L, \eta)$. Any further instantiation will add an additional node and OF -edge to \bar{L}_u and an additional node and F -edge to \bar{R}_u . The universally quantified node (i.e. $qn(u)$) is marked grey. This means that η is only applicable if the grey node is matched to a node with degree (exactly) two. The rule application is performed by calculating the pushout of η (not r) and a valid matching m . The matching is only valid if all edges incident to the grey node have a preimage in \bar{L}_u , such that an application will always result in all incident OF -edges to be replaced by F -edges. Although the number of affected edges can be arbitrary large, the quantification is bounded to the neighbourhood of the grey node and therefore the change is still local.

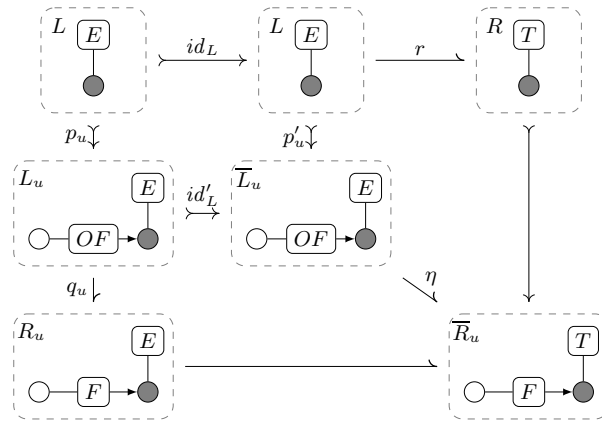


Fig. 3: A possible instantiation of the rule in Figure 1f

The order in which universal quantifications are used to generate instantiations can be neglected, since different sequences will still yield the same instantiation (up to isomorphism). Therefore we can uniquely specify instantiations by the number each universal quantification in its sequence.

Definition 3 (Rule application). *Let ρ be a universally quantified rule. We say that ρ is applicable to a graph G , if there is an instantiation (π, γ) of ρ and a total injective match $m : \bar{L} \rightarrow G$, such that for every $x \in \text{qn}(\rho)$, there is no $e \in E_G$ incident to $m(\pi(x))$ without a preimage in \bar{L} . The application of ρ to G via m results in the graph H , the pushout of m and γ .*

We reuse the notation $G \Rightarrow G'$ to denote a rewriting step from G to G' . The previous definition introduces a restricted form of negative application condition since the existence of an edge, which cannot be mapped by a quantification, may block the application of a rule.

4 A Procedure for Coverability in UGTS

In this paper we focus our attention on verification problems that can be formulated as reachability and coverability decision problems. Given an initial configuration G_0 and a target configuration G_1 reachability consists in checking whether there exists a computation from G_0 to G_1 . The coverability problem is similar to the reachability problem, but additionally relies on an ordering. In this paper we use the subgraph ordering, but there are other suitable orders such as the minor ordering or the induced subgraph ordering [25].

Definition 4 (Subgraph Ordering). *A graph G_1 is a subgraph of G_2 , written $G_1 \subseteq G_2$, if there exists a partial, injective and surjective morphism from G_2 to G_1 , written $\mu : G_2 \rightarrowtail G_1$. Such morphisms are called subgraph morphisms.*

Given a G , a subgraph can always be obtained by a sequence of node and edge deletions. Note that due to the morphism property every edge attached to a deleted node must be deleted as well. Using the subgraph ordering we can represent sets of configurations by minimal graphs and define two variants of the coverability problem.

Definition 5 (Upward Closure). *The upward closure of a set S of graphs is defined as $\uparrow S = \{G' \mid G \subseteq G', G \in S\}$. A set S is upward-closed if it satisfies $S = \uparrow S$. A basis of an upward-closed set S is a set B such that $S = \uparrow B$.*

Definition 6 (Coverability). *Let G_0, G_1 be two graphs. The general coverability problem is to decide whether from G_0 we can reach a graph G_2 such that $G_1 \subseteq G_2$.*

Let \mathcal{G} a set of graphs and let $G_0, G_1 \in \mathcal{G}$. The restricted coverability problem is to decide whether from G_0 we can reach a graph $G_2 \in \mathcal{G}$ such that $G_1 \subseteq G_2$ and every graph on the sequence from G_0 to G_2 is an element of \mathcal{G} .

In other words, a configuration is coverable from some initial configuration if we can reach a configuration containing (as subgraph) a given pattern. Although general

and restricted coverability are both undecidable, we can obtain decidability results by using a backward search introduced for well-structured transition systems [6,22] as already shown in [7]. These systems rely on a *well-quasi-order* (*wqo*), which is a transitive reflexive order \leq such that there is no infinite, strictly decreasing sequence of elements and no infinite antichain, a sequence of pairwise incomparable elements, wrt. \leq . A direct consequence of this property is that every upward-closed set wrt. some wqo has a finite basis. It has been shown that the subgraph ordering is a well-quasi-order on \mathcal{G}_k , the class of graphs in which every undirected path has at most the length k [19]. We remark that the property does not hold if only directed paths are restricted.

The backward search presented in this paper is a version of the general backward search presented in [25] adapted to be compatible with UGTS. We denote the set of *predecessors* for a set of graphs \mathcal{S} by $Pred(\mathcal{S}) = \{G' \mid \exists G \in \mathcal{S}: G' \Rightarrow G\}$. Furthermore we denote the predecessors reachable within multiple step by $Pred^*(\mathcal{S})$ and the *restricted predecessors* by $Pred_{\mathcal{G}}(\mathcal{S}) = Pred(\mathcal{S}) \cap \mathcal{G}$. We will present a procedure for UGTS to compute so-called effective pred-basis and effective \mathcal{G}_k -pred-basis. An *effective pred-basis* for a graph G is a finite basis $pb(G)$ of $\uparrow Pred(\uparrow\{G\})$ and an *effective \mathcal{G}_k -pred-basis* is a finite basis $pb_k(G)$ of $\uparrow Pred_{\mathcal{G}_k}(\uparrow\{G\})$. Using the effective \mathcal{G}_k -pred-basis the backward search will terminate and compute a finite basis \mathcal{B} . If $G \in \uparrow\mathcal{B}$, then G covers a configuration of \mathcal{S} in \Rightarrow (general coverability). If $G \notin \uparrow\mathcal{B}$, then G does not cover a configuration of \mathcal{S} in $\Rightarrow_{\mathcal{G}_k}$ (no restricted coverability), where $\Rightarrow_{\mathcal{G}_k}$ is the restriction $\Rightarrow \cap (\mathcal{G}_k \times \mathcal{G}_k)$. By using the effective pred-basis the backward search computes a finite basis for $Pred^*(\mathcal{S})$, but is not guaranteed to terminate.

The computation of a \mathcal{G}_k -pred-basis is performed by Procedure 1. We assume that for a graph G and a rule ρ there is an upper bound on the length of instantiations necessary to compute a backward step and write $bound_{\rho}(G)$ to denote such an upper bound. The existence of this upper bound is shown later on in Proposition 1. The result of a backward step is a finite set \mathcal{S} of graphs such that $Pred(\uparrow\{G\}) \subseteq \uparrow\mathcal{S}$.

Procedure 1 (Backward Step).

Input: A rule ρ and a graph G .

Procedure:

1. First compute all instantiations $(\pi : L \rightarrow \overline{L}, \gamma : \overline{L} \rightarrow \overline{R})$ of ρ up to the length $bound_{\rho}(G)$.
2. For each γ compute all subgraph morphisms $\mu : \overline{R} \rightarrow R'$. Note that it is sufficient to take a representative R' for each of the finitely many isomorphism classes.
3. For each $\mu \circ \gamma$ compute all total injective morphisms $m' : R' \rightarrow G$ (co-matches of R' in G).
4. For each such morphism m' calculate all minimal pushout complements $G', m : \overline{L} \rightarrow G'$ of m' and $\mu \circ \gamma$ where m is injective and G' is an element of \mathcal{G}_k . Drop all G' where m does not satisfy the application condition of Definition 3, i.e. there is an edge incident to a quantified node which is not in the matching.

Result: The set of all graphs not dropped in Step 4, written $pb_k(G)$.

The motivation behind Step 2 is that G represents not just itself but also its upward closure. Therefore, the rule must also be applied to every graph larger than G . Instead of using partial co-matches we concatenate with subgraph morphisms to simulate this behaviour.

The procedure for a single backward step can be used to define a backward search procedure for the coverability problem for UGTS. The procedure exploits the property that, even if compatibility is not satisfied, $Pred(\uparrow\mathcal{S}) \subseteq \uparrow Pred(\uparrow\mathcal{S})$ still holds for every set of graphs \mathcal{S} . We can iteratively compute backward steps for all minimal graphs G of $\uparrow\mathcal{S}$ and check that no initial state is reached backwards.

Procedure 2 (Backward Search).

Input: A natural number k , a set \mathcal{R} of graph transformation rules and a finite set of final graphs \mathcal{F} . Start with the working set $\mathcal{W} = \mathcal{F}$.

Backward Step: For each $G \in \mathcal{W}$ add all graphs of $pb_k(G)$ to \mathcal{W} and minimize \mathcal{W} by removing all graphs H' for which there is a graph $H'' \in \mathcal{W}$ with $H' \neq H''$ and $H'' \subseteq H'$. Repeat this backward steps until the sequence of working sets \mathcal{W} becomes stationary, i.e. for every $G \in \mathcal{W}$ the computation of the backward step using G results in no change of \mathcal{W} .

Result: The resulting set \mathcal{W} contains minimal representatives of graphs from which a final state is coverable. This set may be an over-approximation, even without quantified rules.

To show the termination of Procedure 1 and 2 it is important to show the existence of a bounding function $bound_\rho()$. By the following proposition this function exists for every rule ρ , but as we will show later this bound can be tightened in most cases.

Proposition 1. *Let ι be an instantiation of length k of some rule ρ . If k is larger than the number of nodes and edges of G , then every graph computed by the backward application of ι is already represented by the backward application of an instantiation of lower length.*

The following two lemmas prove that Procedure 1 computes a finite basis of an over-approximation of the restricted predecessors.

Lemma 1. *The set $pb_k(G)$ is a finite subset of $Pred(\uparrow\{G\})$ and $pb_k(G) \subseteq \mathcal{G}_k$.*

Lemma 2. *It holds that $\uparrow pb_k(G) \supseteq \uparrow Pred_{\mathcal{G}_k}(\uparrow\{G\})$.*

We recapitulate our main result in the following proposition.

Proposition 2. *For each graph G , $pb_k(G)$ is an effective \mathcal{G}_k -pred-basis. Furthermore, Procedure 2 terminates and computes an over-approximation of all configurations in \mathcal{G}_k from which a final configuration is coverable.*

Proof. By Lemma 1 and 2 we know that $\uparrow pb_k(G) = \uparrow Pred_{\mathcal{G}_k}(\uparrow\{G\})$ and thus $pb_k(G)$ is a \mathcal{G}_k -pred-basis. According to Proposition 1 for every $\rho \in \mathcal{R}$ the number of necessary instantiation steps is bounded by $bound_\rho(G)$, thus, the number of instantiations is fine. For each instantiation the minimal pushout complements restricted to \mathcal{G}_k are finite and computable. Since the subgraph ordering is decidable the minimization is computable and $pb_k(G)$ is effective.

Since the subgraph ordering is a wqo on \mathcal{G}_k , every infinite increasing sequence of upward-closed set becomes stationary. The upward-closures of the working sets \mathcal{W} form such an infinite increasing sequence, thus the termination criteria of Procedure 2 will be satisfied at some point. \square

A Variant of $pb_k()$ Without Path Bound

In Step 4 of Procedure 1 every graph which is not an element of \mathcal{G}_k is dropped. This is needed to guarantee that the working set of Procedure 2 becomes stationary and the search terminates. However, this restriction can be dropped to obtain a backward search which solves the general coverability problem. Termination is not guaranteed, but correctness can be proven analogously to the restricted variant, as already shown in [25]. Let $pb()$ be Procedure 1 without the restriction to \mathcal{G}_k . We summarize the decidability of this second variant in the following proposition.

Proposition 3. *For each graph G , $pb(G)$ is an effective pred-basis. Furthermore, when using $pb()$ instead of $pb_k()$, Procedure 2 computes an over-approximation of all configurations from which a final configuration is coverable.*

Experimental Results

We added support for universally quantified rules to the UNCOVER tool. This tool can perform the backward search for the subgraph ordering and the minor ordering (a coarser order compared to subgraphs). Both variants of the backward search are implemented, but a timeout might occur when using the unrestricted variant. However, given the rules in Figure 1 and the error graphs in Figure 4 the unrestricted variant terminates after 12 seconds and results in a set of 12 minimal graphs. Two of these graphs are the initial error graphs and two other computed graphs are shown in Figure 5. Every minimal graph contains a node in the state E . Since initially no philosopher is eating, the initial configuration is not represented and none of the initial error graphs is reachable. This proves that two adjacent philosophers cannot be eating at the same time.

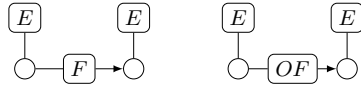


Fig. 4: Two error configurations in the Dining Philosophers Problem

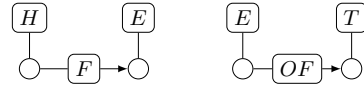


Fig. 5: Two other error graphs computed by the backward search

5 Optimizations

In this section we discuss and formalize some optimizations that can be applied to the basic backward procedure described in the previous section.

Lifting the Application Condition to a Post Conditions In Procedure 1 the application condition is checked in Step 4 for each pushout complement. However, by lifting the application condition over the instantiation we can check beforehand whether the backward step yields new graphs. We show the lifting in the following lemma.

Lemma 3. *Let ρ be a rule, $(\pi : L \rightarrow \overline{L}, \gamma : \overline{L} \rightarrow \overline{R})$ an instantiation of ρ and $m : \overline{R} \rightarrow G$ a co-match of the instantiation to some graph G . If there is a node $x \in \text{qn}(\rho)$ where $m(\gamma(\pi(x)))$ is defined and attached to an edge e without preimage in \overline{R} , then there is no pushout complement H of γ, m satisfying the condition of Definition 3.*

Tightening the Upper Bound of Instantiations The bound on the length of instantiations proven to exist in Proposition 1 can be improved depending on the rule used. Let $\rho = (r : L \rightarrow R, U)$ be a rule. Obviously $\text{bound}_\rho(G) = 0$ if $U = \emptyset$. The same holds if instantiations only increase the left side of the rule, i.e. for every $u \in U$ given the instantiation $(\text{id}_L, r) \diamond u = (\pi : L \rightarrow \overline{L}_u, \gamma : \overline{L}_u \rightarrow \overline{R}_u)$, the graphs \overline{R}_u and R are isomorphic.

A more common situation is that quantifications do not add edges to the right side of the instantiations which are solely incident to nodes of the original rule r . This is case for all rules used in Example 1. The bound can be reduced as shown below.

Lemma 4. *Let $\rho = (r : L \rightarrow R, U)$ and let $(\text{id}_L, r) \diamond u = (\pi : L \rightarrow \overline{L}_u, \gamma : \overline{L}_u \rightarrow \overline{R}_u)$. If for every $u \in U$ every edge $e \in \overline{R}_u$ without preimage in R is connected to a node $v \in \overline{R}_u$ without preimage in R , then $\text{bound}_\rho(G) = |V_G|$.*

Optimization by Preparation The general framework in [25] uses a preparation step in the backward search to compute the concatenation of rules and subgraph morphisms performed in Step 2 of Procedure 1. This is not fully possible with universally quantified rules since the instantiations are generated within the backward steps. However, the preparation step can be performed for rules without universal quantifications. For rules with quantification the inner rule morphism can be concatenated with subgraph morphisms to partially prepare the rule. It can also be show that any concatenation of an instantiation and a subgraph morphism which is also a subgraph morphism, will not yield new graph in the backward step and thus can be dropped. This also holds for rules with universal quantification if all possible instantiations are also subgraph morphisms.

6 Conclusions and Related Work

In this paper we introduced a categorical formalization for an extension of graph transformation systems with universally quantified rules built on the single pushout approach. These rules are powerful enough to model distributed algorithms which use broadcast communication. A similar concept are adaptive star grammars [20] where the left-hand side of a rule is a star, i.e. a designated center node connected to a set of other nodes. Arbitrary large graphs can be matched by cloning parts of the star, which is – apart of the restriction to stars – one of the main differences to our approach. Technically our instantiations are a special form of amalgamated graph transformations [9], a technique to merge rules.

The backward search procedure presented in this paper is an extension of [25] with universally quantified rules and can be used for the verification of distributed algorithms, similar to [14]. There the induced subgraph ordering was used, which was also

shown to be compatible with the framework in [25]. However, our quantifications differ as we have a stronger negative application condition such that the induced subgraph ordering is not enough to cause our UGTS to satisfy the compatibility condition. This also causes the approach to differ in expressiveness. In general our approach should be compatible with the induced subgraph ordering and the minor ordering, but we did not yet investigate this.

Parameterized verification of combinations of automata- and graph-based models of distributed systems has been studied, e.g. in [10,4,15,16,13,12]. In [5] we applied graph-based transformations to model intermediate evaluations of non-atomic mutual exclusion protocols with universally quantified conditions. The conditions are not defined however in terms of graph rewrite rules. Semi-decision procedures can be defined by resorting to upward closed abstractions during backward search (monotonic abstraction as in [11]). In [10] we studied decidability of reachability and coverability for a graph-based specification used to model biological systems. Among other results, we proved undecidability for coverability for graph rewrite systems that can only increase the size of a configuration. Reachability problems for graph-based representations of protocols have also been considered in [4] where symbolic representations combining a special graph ordering and constraint-based representation of relations between local data of different nodes have been used to verify parameterized consistency protocols. Coverability for GTS is studied in [8] where it was proved that it is decidable for bounded path graphs ordered via subgraph inclusion. A model with topologies represented as acyclic directed graphs has been presented in [1]. Coverability for automata-based models of broadcast communication has recently been studied in [15,16,13,18,12]. In the context of program analysis approximated backward search working on graphs representing data structures with pointers have been considered in [2]. In this setting approximations are defined via edges or node deletion.

References

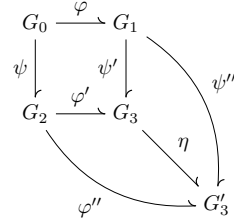
1. P. A. Abdulla, M. F. Atig, and O. Rezine. Verification of directed acyclic ad hoc networks. In *FMOODS/FORTE*, pages 193–208, 2013.
2. P. A. Abdulla, J. Cederberg, and T. Vojnar. Monotonic abstraction for programs with multiply-linked structures. *Int. J. Found. Comput. Sci.*, 24(2):187–210, 2013.
3. P. A. Abdulla, G. Delzanno, and A. Rezine. Approximated parameterized verification of infinite-state processes with global conditions. *Formal Methods in System Design*, 34(2):126–156, 2009.
4. P. A. Abdulla, G. Delzanno, and A. Rezine. Automatic verification of directory-based consistency protocols with graph constraints. *Int. J. Found. Comput. Sci.*, 22(4), 2011.
5. P. A. Abdulla, N. Ben Henda, G. Delzanno, and A. Rezine. Handling parameterized systems with non-atomic global conditions. In *VMCAI’08*, volume 4905 of *LNCS*, pages 22–36. Springer, 2008.
6. P. A. Abdulla, K. Čerāns, B. Jonsson, and Y. Tsay. General decidability theorems for infinite-state systems. In *Proc. of LICS ’96*, pages 313–321. IEEE, 1996.
7. N. Bertrand, G. Delzanno, B. König, A. Sangnier, and J. Stückrath. On the decidability status of reachability and coverability in graph transformation systems. In *RTA’12*, volume 15 of *LIPICs*, pages 101–116. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

8. N. Bertrand, G. Delzanno, B. König, A. Sangnier, and J. Stückrath. On the decidability status of reachability and coverability in graph transformation systems. In *RTA*, pages 101–116, 2012.
9. P. Boehm, H. Fonio, and A. Habel. Amalgamation of graph transformations: A synchronization mechanism. *Journal of Computer and System Sciences*, 34:377 – 408, 1987.
10. G. Delzanno, C. Di Giusto, M. Gabbriellini, C. Laneve, and G. Zavattaro. The κ -lattice: Decidability boundaries for qualitative analysis in biological languages. In *CMSB*, pages 158–172, 2009.
11. G. Delzanno and A. Rezine. A lightweight regular model checking approach for parameterized systems. *STTT*, 14(2):207–222, 2012.
12. G. Delzanno, A. Sangnier, and R. Traverso. Parameterized verification of broadcast networks of register automata. In *RP’13*, pages 109–121, 2013.
13. G. Delzanno, A. Sangnier, R. Traverso, and G. Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *FSTTCS’12*, volume 18 of *LIPICs*, pages 289–300. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
14. G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *Proc. CONCUR ’10*, pages 313–327. Springer, 2010. LNCS 6269.
15. G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR’10*, volume 6269 of *LNCS*, pages 313–327. Springer, 2010.
16. G. Delzanno, A. Sangnier, and G. Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *FOSSACS’11*, volume 6604 of *LNCS*, pages 441–455. Springer, 2011.
17. G. Delzanno and J. Stückrath. Parameterized verification of graph transformation systems with whole neighbourhood operations. In *RP’14*, 2014.
18. G. Delzanno and R. Traverso. Decidability and complexity results for verification of asynchronous broadcast networks. In *LATA*, pages 238–249, 2013.
19. G. Ding. Subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 16:489–502, November 1992.
20. F. Drewes, B. Hoffmann, D. Janssens, M. Minas, and N. V. Eetvelde. Adaptive star grammars. In *Proc. of ICGT ’06 (International Conference on Graph Transformation)*, pages 77–91. Springer, 2006. LNCS 4178.
21. H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, and A. Corradini. Algebraic approaches to graph transformation—part II: Single pushout approach and comparison with double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, chapter 4. World Scientific, 1997.
22. A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, April 2001.
23. M. Heumüller, S. Joshi, B. König, and J. Stückrath. Construction of pushout complements in the category of hypergraphs. In *Proc. of GCM ’10 (Workshop on Graph Computation Models)*, 2010.
24. S. Joshi and B. König. Applying the graph minor theorem to the verification of graph transformation systems. In *Proc. of CAV ’08*, pages 214–226. Springer, 2008. LNCS 5123.
25. Barbara König and Jan Stückrath. A general framework for well-structured graph transformation systems. In P. Baldan and D. Gorla, editors, *Proc. of CONCUR 2014*, volume 8704 of *LNCS*, pages 467–481. Springer, 2014.
26. K. S. Namjoshi and R. J. Treffer. Uncovering symmetries in irregular process networks. In Roberto Giacobazzi, Josh Berdine, and Isabella Mastroeni, editors, *VMCAI*, volume 7737 of *Lecture Notes in Computer Science*, pages 496–514. Springer, 2013.

A Important Properties of Pushouts

Pushouts and pushout complements are a well-known notion in category theory and are the basis for the single pushout approach as well as the double pushout approach [21]. We briefly recall the definition of a pushout and a few important properties of such.

Definition 7. Let $\varphi: G_0 \rightarrow G_1$ and $\psi: G_0 \rightarrow G_2$ be two partial graph morphisms. The pushout of φ and ψ consists of a graph G_3 and two morphisms $\psi': G_1 \rightarrow G_3$, $\varphi': G_2 \rightarrow G_3$ such that $\psi' \circ \varphi = \varphi' \circ \psi$ and for every other pair of morphisms $\psi'': G_1 \rightarrow G'_3$, $\varphi'': G_2 \rightarrow G'_3$ such that $\psi'' \circ \varphi = \varphi'' \circ \psi$ there exists a unique morphism $\eta: G_3 \rightarrow G'_3$ with $\eta \circ \psi' = \psi''$ and $\eta \circ \varphi' = \varphi''$.

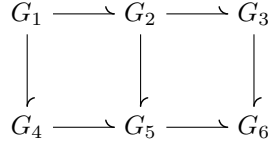


Without proof we use the following properties of pushouts in our proofs in Appendix B.

Lemma 5. Let $\varphi: G_1 \rightarrow G_2$ and $\psi: G_1 \rightarrow G_3$ be morphisms and let $\varphi': G_3 \rightarrow G_4$, $\psi': G_2 \rightarrow G_4$ be their pushout. If ψ is total and injective, then ψ' is also total and injective.

Lemma 6. Let morphisms as shown in the diagram below be given. It can be shown that the following two properties hold for any category.

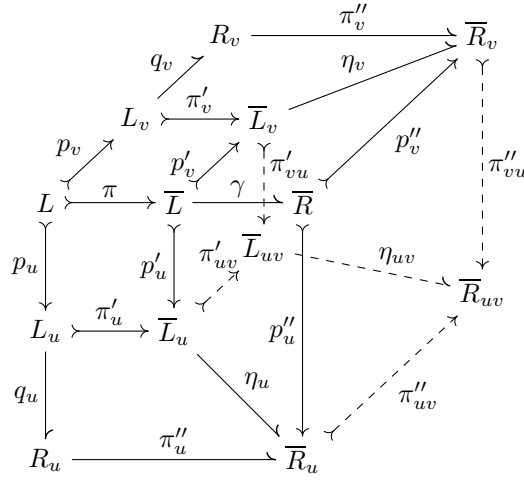
1. If the left and the right squares below are pushouts, the outer square is a pushout as well.
2. If the left square and the outer square are pushouts, the right square is a pushout as well.



B Proofs

Lemma 7. Let $\rho = (r, U)$ be a rule and let $f: U \rightarrow \mathbb{N}_0$ be any function assigning a quantity to each universal quantification. Every instantiation of ρ which is generated by using $f(u)$ occurrences for each u respectively, yields the same morphisms (up to isomorphism).

Proof. We show this property by showing that we can swap each two instantiation steps without changing the instantiation containing both steps. Let $\iota = (\pi, \gamma)$ be an instantiation of some rule $\rho = (r, U)$ and let $u = (p_u, q_u), v = (p_v, q_v) \in U$ be two universal quantifications as shown in the diagram below. There the upper part of the diagram is the instantiation $\iota_v = \iota \diamond v$, while the front part of the diagram is the instantiation $\iota_u = \iota \diamond u$.



Let \bar{L}_{uv} be the pushout of p'_u, p'_v and let \bar{R}_{uv} be the pushout of p''_u, p''_v . By the properties of pushouts a unique η_{uv} exists and we will show that $\iota_u \diamond v = (\pi'_{vu} \circ \pi'_v \circ p_v, \eta_{uv}) = \iota_v \diamond u$.

By construction all squares $p'_u \circ \pi = \pi'_u \circ p_u, p'_v \circ \pi = \pi'_v \circ p_v$ and $\pi'_{vu} \circ p'_v = \pi'_{uv} \circ p'_u$ are pushouts. Therefore, the squares $\pi'_{vu} \circ \pi'_v \circ p_v = \pi'_{uv} \circ p'_u \circ \pi$ and $\pi'_{vu} \circ p'_v \circ \pi = \pi'_{uv} \circ \pi'_u \circ p_u$ are pushouts as well. Thus, \bar{L}_{uv} is the pushout of $p_v, p'_u \circ \pi$ computed in the construction of $\iota_u \diamond v$ as well as the pushout of $p_u, p'_v \circ \pi$ computed in the construction of $\iota_v \diamond u$. The same property holds for \bar{R}_{uv} using the three large outer squares. Since η_{uv} is unique, both sequences of the instantiation steps give rise to the same morphisms. This means that every instantiation can be uniquely characterized only by the number on instantiation steps for each $u \in U$. \square

Lemma 8. *Let $\rho = (r, U)$ be a rule and let $(\pi : L \rightarrow \bar{L}, \gamma : \bar{L} \rightarrow \bar{R})$ be an instantiation of ρ . For every further instantiation $(\pi, \gamma) \diamond u$ using some $u \in U$, there are two subgraph morphisms $\mu'_u : \bar{L}_u \rightarrow \bar{L}$ and $\mu''_u : \bar{R}_u \rightarrow \bar{R}$ such that $\gamma \circ \mu'_u = \mu''_u \circ \eta$.*

Proof. By definition p_u and $q_u \circ p_u$ are total and injective, thus, p'_u and $q''_u \circ p'_u$ are total and injective as well (see Lemma 5). Hence, the reverse morphisms μ'_u and μ''_u are partial, injective and surjective, i.e. subgraph morphisms. By using Lemma 6 it can be shown that by forming the pushout \bar{R}'_u of p'_u and γ , the pushout R'_u of q_u and π' and then the pushout \bar{R}_u of q'_u and γ' , we obtain the same graph \bar{R}_u as by forming the pushout of $q_u \circ p_u$ and $\gamma \circ \pi$ directly. Furthermore, the diagram below commutes with the exception of μ'_u and μ''_u , for which we still have to show the commutativity with η and γ .

$$\begin{array}{ccccc}
L & \xrightarrow{\pi} & \bar{L} & \xrightarrow{\gamma} & \bar{R} \\
p_u \downarrow & & p'_u \downarrow & \mu'_u & p''_u \downarrow \\
L_u & \xrightarrow{\pi'} & \bar{L}_u & \xrightarrow{\gamma'} & \bar{R}' \\
q_u \downarrow & & q'_u \downarrow & \eta & q''_u \downarrow \\
R_u & \xrightarrow{\pi''} & R'_u & \xrightarrow{\gamma''} & \bar{R}_u
\end{array}
\quad \begin{array}{c} \curvearrowright \mu''_u \end{array}$$

Let $x \in \bar{L}_u$ and assume $\gamma(\mu'_u(x))$ is defined. This means that there is exactly one $x' \in \bar{L}$ with $p'_u(x') = x$ and $\gamma(x')$ is defined. Since $q''_u \circ p''_u$ is total and injective, there is an $x'' \in \bar{R}_u$ with $q''_u(p''_u(\gamma(x'))) = x''$. Due to commutativity of the diagram we obtain $\eta(x) = x''$. Hence, we know that $\mu''_u(\eta(x))$ is defined and $\gamma(\mu'_u(x)) = \mu''_u(\eta(x))$.

Now assume $\mu'_u(x)$ is defined, but $\gamma(\mu'_u(x))$ is undefined. Because of commutativity, $\gamma'(x)$ is undefined as well and therefore also $\eta(x) = q''_u(\gamma'(x))$ is undefined. Thus, $\gamma(\mu'_u(x)) = \mu''_u(\eta(x))$ are both undefined.

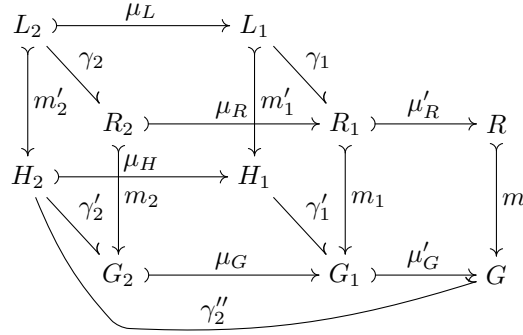
Now assume $\mu'_u(x)$ is undefined. If $\eta(x)$ is undefined, $\gamma(\mu'_u(x)) = \mu''_u(\eta(x))$ are obviously both undefined, so we assume that $\eta(x)$ is defined and show that $\eta(x)$ has no preimages under $q''_u \circ p''_u$. For this we only have to consider elements in \bar{R}' which have preimages in \bar{L}_u , since an element without a preimage and mapped to $\eta(x)$ would violate the pushout property of the lower right square. We observe that $\gamma'(x)$ has no preimage in \bar{R} , since the top right square would not be a pushout. In fact this holds for every $x' \in \bar{L}_u$ with $\eta(x') = \eta(x)$ if x' has no preimage in \bar{L} . By the same argument we also know that $\gamma'(x')$ has exactly one preimage in \bar{L}_u . This means that two x' with and without preimage in \bar{L} are not merged by γ' . By showing that these x' are also not merged by q'_u , we know that their image in the pushout \bar{R}_u would not be equal and prove that there are in fact no x' with preimage in \bar{L} .

If x' has a preimage in \bar{L} but not in L_u , then x' is not merged with any other element by q'_u , since the left lower square is a pushout. If x' has a preimage in \bar{L} and L_u , it also has (exactly) one preimage in L , because of the top left square being a pushout. Thus, by Definition 1 q_u may not merge the preimage of x' with anything else, especially not with the preimage of x . Since neither π' nor q_u merge the preimage of x' with anything, x is not merged with anything via q'_u as well. Thus, $\eta(x') = \eta(x)$ cannot hold and $\eta(x)$ has no preimage in \bar{R} .

We have shown that $\gamma(\mu'_u(x))$ is undefined if and only if $\mu''_u(\eta(x))$ is undefined, thus the commutativity $\gamma \circ \mu'_u = \mu''_u \circ \eta$ follows from μ'_u being the reverse of p'_u and μ''_u being the reverse of $q''_u \circ p''_u$. \square

Lemma 9. Let $\rho = (r, U)$ be a rule and let $(\pi_i : L \rightarrowtail L_i, \gamma_i : L_i \rightarrow R_i)$ for $i \in \{1, 2\}$ be two instantiations of ρ with $(\pi_2, \gamma_2) = (\pi_1, \gamma_1) \diamond u$ for some $u \in U$. Furthermore, let $\mu_L : L_2 \rightarrowtail L_1$, $\mu_R : R_2 \rightarrowtail R_1$, $\mu'_R : R_1 \rightarrowtail R$ be subgraph morphisms with $\gamma_1 \circ \mu_L = \mu_R \circ \gamma_2$ and let $m : R \rightarrowtail G$ be a match. For every pushout complement H_2 of $\mu'_R \circ \mu_R \circ \gamma_2$ and m where $m'_2 : L_2 \rightarrowtail H_2$ is total and injective, there is a pushout complement H_1 of $\mu'_R \circ \gamma_1$ and m with $H_1 \subseteq H_2$.

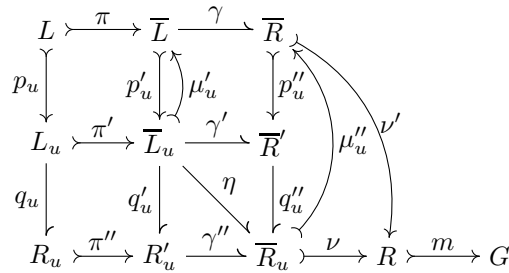
Proof. We will show this by using the fact, that subgraph morphisms are preserved by total pushouts and successively building the commuting diagram below.



Let H_2 be a pushout complement of $\mu'_R \circ \mu_R \circ \gamma_2$ and m , where m'_2 is total and injective. We compute the pushout G_2 of γ_2 and m'_2 and then the pushout G_1 of μ_R and m_2 . Due to Lemma 6, G_1 is also the pushout of $\mu_R \circ \gamma_2$ and m'_2 and therefore there is a unique $\mu'_G : G_1 \rightarrowtail G$ such that the diagram commutes. Since G is the pushout of $\mu'_R \circ \mu_R \circ \gamma_2$ and m'_2 , the rightmost square is in fact a pushout as well. Now form the pushout H_1 of μ_L and m'_2 . Again the existence of γ'_1 follows from the pushout properties. Since the diagram $m_1 \circ \gamma_1 \circ \mu_L = \gamma'_1 \circ \mu_H \circ m'_2$ commutes with the pushout $m_1 \circ \mu_R \circ \gamma_2 = \mu_G \circ \gamma'_2 \circ m'_2$, it is also a pushout and hence, G_1 is a pushout of γ_1 and m'_1 . This means that H_1 is in fact a pushout complement of $\mu'_R \circ \gamma_1$ and m . Since subgraph morphisms are preserved by total pushouts, μ_H is a subgraph morphism. Thus, $H_1 \subseteq H_2$. \square

Proposition 1. *Let ι be an instantiation of length k of some rule ρ . If k is larger than the number of nodes and edges of G , then every graph computed by the backward application of ι is already represented by the backward application of an instantiation of lower length.*

Proof. Let $\iota_{k-1} = (\pi : L \rightarrowtail \bar{L}, \gamma : \bar{L} \rightarrow \bar{R})$ be a rule instantiation of length $k - 1$ of (r, U) such that $\iota_k = (\pi, \gamma) \diamond u$ for some $u \in U$, let $\nu : \bar{R}_u \rightarrowtail R$ be a subgraph morphism and let $m : R \rightarrowtail G$ be a co-match as shown in the diagram below.



From Lemma 8 we know that μ'_u and μ''_u exist and the diagram commutes. We will show the existence of a subgraph morphism $\nu' : \bar{R} \rightarrowtail R$ satisfying $\nu = \nu' \circ \mu''_u$. Then from Lemma 9 it follows that every graph computed by a backward step of $\nu \circ \eta$, the instantiation ι_k , is already represented by a backward step of $\nu' \circ \gamma$, the instantiation ι_{k-1} .

First assume that $\gamma''(\pi''(x_u))$ is undefined for every $x_u \in R_u$ which has no preimage under $q_u \circ p_u$. We can show that $q''_u \circ p''_u$ is a subgraph morphism by showing that it is surjective. Assume there is an $\bar{x}_u \in \bar{R}_u$ without preimage under $q''_u \circ p''_u$. Since the large square is a pushout, there is an $x'_u \in R_u$ with $\gamma''(\pi''(x'_u)) = \bar{x}_u$. By the first assumption x'_u must have a preimage $x \in L$ under $q_u \circ p_u$ for $\gamma''(\pi''(x'_u))$ to be defined. Due to the commutativity, $\gamma(\pi(x))$ is defined and there is a preimage of \bar{x}_u in \bar{R} , violating the second assumption. Hence, $q''_u \circ p''_u$ is a subgraph morphism commuting with μ''_u (in fact \bar{R} and \bar{R}_u are isomorphic). The morphism $\nu' = \nu \circ q''_u \circ p''_u$ satisfies the necessary properties.

If at least one quantification within ι_k satisfies the previous restriction, by Lemma 7 we can assume w.l.o.g. that it is the last quantification step. So assume for every quantification step there is at least one $x_u \in R_u$ without preimage under $q_u \circ p_u$ such that $\gamma''(\pi''(x_u))$ is defined. Since $\gamma''(\pi''(x_u))$ has no preimage under $q''_u \circ p''_u$ (otherwise it would have a preimage in L), the graph \bar{R}_u has at least k nodes and edges. Thus, since R has less than k nodes and edges, for at least one instantiation step within ι_k for every $x'_u \in R_u$ without a preimage under $q_u \circ p_u$, the image $\nu(\gamma''(\pi''(x'_u)))$ is undefined. Again by Lemma 7 we can assume w.l.o.g. that it is the last quantification of ι_k .

In this case $\nu' = \nu \circ q''_u \circ p''_u$ satisfies the necessary conditions. Obviously ν' is injective and $\nu = \nu' \circ \mu''_u$ holds, so it remains to be shown that it is surjective. Assume there is an $y \in R$ without a preimage under ν' . Since ν is injective and surjective, there is exactly one $\bar{y}_u \in \bar{R}_u$ with $\nu(\bar{y}_u) = y$. Because of commutativity, \bar{x}_u cannot have a preimage under $q''_u \circ p''_u$. Since the outer square is a pushout, there has to be an $y_u \in R_u$ with $\gamma''(\pi''(y_u)) = \bar{y}_u$. By assumption this y_u has a preimage under $q_u \circ p_u$ (otherwise $\nu(\gamma''(\pi''(y_u)))$ would be undefined), which in turn has an image in \bar{R} . By commutativity y must have a preimage under ν' . Thus, ν' is surjective and a subgraph morphism. \square

Lemma 10. *Let $r : L \rightarrowtail R$ be a partial morphism and let $m : R \rightarrowtail G$ be total and injective. The set of pushout complements G' of m and r where $m' : L \rightarrowtail G'$ is injective has finitely many minimal elements and this minimal elements are computable.*

Proof. In [24] it was shown how the minimal pushout complements with respect to the minor ordering and conflict-free matching can be computed. The procedure can be easily adapted to subgraphs with injective matching. First the co-match m can be assumed to injective, since a non-injective co-match implies a non-injective match m' (see Lemma 5). Furthermore a pushout complement with a non-injective m' can be dropped since neither it nor any larger pushout complement has an injective match, as shown below.

$$\begin{array}{ccccc}
 & & L & \xrightarrow{\gamma} & R \\
 & m'' \swarrow & \downarrow m' & & \downarrow m \\
 H' & \xrightarrow{\mu} & H & \xrightarrow{\gamma'} & G
 \end{array}$$

Assume there are two pushout complements H, H' with $H \subseteq H'$ and m' is non-injective. There is a subgraph morphism $\mu : H' \rightarrowtail H$ such that the diagram above

commutes, especially $m' = \mu \circ m''$ holds. Since m' is non-injective but μ is injective, m'' must be non-injective as well. Thus H' can be dropped as well.

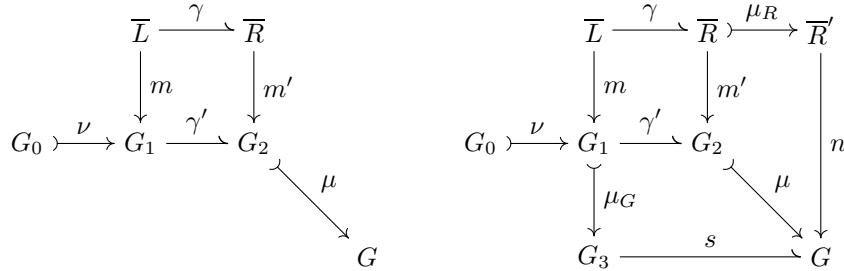
Since we obtain a subset of the minimal pushout complements of [24], the finiteness of this set is preserved. \square

Lemma 1. *The set $pb_k(G)$ is a finite subset of $Pred(\uparrow\{G\})$ and $pb_k(G) \subseteq \mathcal{G}_k$.*

Proof. We have proven this statement in [25] for rules without universal quantification and for conflict-free matches. The proof can be directly extended to this setting by using the fact that every injective match is automatically conflict-free. Since the number of instantiations is bounded by $bound_\rho(S)$ for every rule ρ and the number of subgraph morphisms $\mu : \overline{R} \rightarrow R'$ is finite (up to isomorphism), the number of morphisms for which the pushout complement need to be computed is finite as well. Furthermore by Lemma 10 the set of minimal pushout complements is finite and computable, thus $pb_k(G)$ is a finite sets. \square

Lemma 2. *It holds that $\uparrow pb_k(G) \supseteq \uparrow Pred_{\mathcal{G}_k}(\uparrow\{G\})$.*

Proof. Let G_0 be an element of $\uparrow Pred_{\mathcal{G}_k}(\uparrow\{G\})$. Then there is a minimal representative $G_1 \in Pred(\uparrow\{G\})$ with $G_1 \subseteq G_0$ via some morphism $\nu : G_0 \rightarrow G_1$ and an instantiation $(\pi : L \rightarrow \overline{L}, \gamma : \overline{L} \rightarrow \overline{R})$ of some rule ρ rewriting G_1 with a injective match m satisfying the application conditions of Definition 3 to some element G_2 of $\uparrow\{G\}$. In [25] it was shown that subgraph morphisms are pushout closed. Since m is injective and therefore conflict-free, the left diagram below can be extended to the right diagram below, where the inner and outer squares are pushouts.



Since m and μ_G are injective, $\mu_G \circ m$ is injective as well and because of Lemma 5 we know that n is also injective. Furthermore the pushout closure guarantees that $\mu_G \circ m$ is total. Since m satisfied the application condition, every edge in G_1 incident to a universally quantified node has a preimage in \overline{L} and therefore also an image in G_3 . The surjectivity of μ_G ensures that the application condition is also satisfied by $\mu_G \circ m$. Note that since G_1 is an element of \mathcal{G}_k and \mathcal{G}_k is downward-closed, G_3 is also in \mathcal{G}_k .

Since the outer square is a pushout, G_3 is a pushout complement object. Thus, a graph G_4 with $\mu'_G : G_4 \subseteq G_3$ will be obtained by the procedure $pb_k()$ in Step 4 using the instantiation $\mu_R \circ \gamma$. By the same argument as above $\mu'_G \circ \mu_G \circ m$ satisfies the application condition and is an element of \mathcal{G}_k , thus G_4 will not be dropped by the procedure. Summarized, this means that $pb_k()$ computes a graph G_4 for every graph G_0 such that $G_4 \subseteq G_3 \subseteq G_1 \subseteq G_0$, i.e. every G_0 is represented by an element of $pb_k(G)$. \square

Lemma 3. *Let ρ be a rule, $(\pi : L \rightarrow \bar{L}, \gamma : \bar{L} \rightarrow \bar{R})$ an instantiation of ρ and $m : \bar{R} \rightarrow G$ a co-match of the instantiation to some graph G . If there is a node $x \in \text{qn}(\rho)$ where $m(\gamma(\pi(x)))$ is defined and attached to an edge e without preimage in \bar{R} , then there is no pushout complement H of γ, m satisfying the condition of Definition 3.*

Proof. Assume there is a $x \in \text{qn}(u)$ where $x' = m(\gamma(\pi(x)))$ is defined and there is an edge e attached to x' without preimage in \bar{R} . Furthermore, assume H with $m' : \bar{L} \rightarrow H$ and $\gamma' : H \rightarrow G$ is a pushout complement of γ, m . Since the diagram is a pushout, there is an $e' \in H$ with $\gamma'(e') = e$, otherwise the mediating morphism does not exist or is not unique. By commutativity of the diagram, e' is attached to $m'(\pi(x))$ and there cannot be an $e'' \in \bar{L}$ with $m'(e'') = e'$. Since $x \in \text{qn}(u)$, this violates the condition of Definition 3. \square

Lemma 4. *Let $\rho = (r : L \rightarrow R, U)$ and let $(id_L, r) \diamond u = (\pi : L \rightarrow \bar{L}_u, \gamma : \bar{L}_u \rightarrow \bar{R}_u)$. If for every $u \in U$ every edge $e \in \bar{R}_u$ without preimage in R is connected to a node $v \in \bar{R}_u$ without preimage in R , then $\text{bound}_\rho(G) = |V_G|$.*

Proof. This can be shown by using the proof ideas of Proposition 1. If ν does not delete all elements of \bar{R}_u which were created in the instantiation step, \bar{R}_u contains at least one node more than R . If the created element not deleted by ν is an edge, by conditions of this lemma, it is incident to a created node not deleted by ν . Thus, every non-negligible instantiation step increases the number of nodes of the right side by at least one. No matchings can exist if the number of instantiation steps is larger than the number of nodes in G . \square